Before beginning the coursework, it is important to recognise what the primary problems to be solved are. The solutions to these problems then motivate the structure of the solution.

The first challenge is to recognise points in the task where the built-in C types are insufficient. The most obvious situation where this is the case is how to handle complex numbers which are necessary for Fourier transforms. This problem was solved using a struct which represents a complex number by keeping track of the real and imaginary parts using two doubles. This is a convenient solution as it allows for arrays of complex numbers to be handled easily and used to perform the discrete Fourier transform (DFT) and inverse DFT (IDFT). Additionally, this allows the functions h1 and h2 to return a single variable comprising both the real and imaginary parts. The other, less obvious, case for using a struct is when reading in the data from the file “h3.txt”. By creating a struct to store an index position, time value, and complex number, each row can be read ensuring the data stays localised in a way which makes more sense than storing them in separate arrays. These structs are as follows:

The second challenge is the implementation of the DFT & IDFT equations provided in the question sheet. F

----------------

Before talking about the details of the different parts of question 3, some details about the overall structure of the program. In general, if a particular process is may be required multiple times during question 3 it has been converted into a function which handles this to improve readability. For example, the “writeComplex” function which writes time and complex number arrays to a specified file. Two custom structs are used to provide complex number functionality and to read in the data from the provided data file “h3.txt”, named Complex and Measurement respectively. Finally, to connect details mentioned here with the actual implementation, each part of question 3 has been turned into its own function when there is ‘work’ which needs to be done. For example, 3.a does not have its own function but 3.b does. These functions are then called in the main function, this allows the logic that led to the implementation to remain clearer.